COL 352 Lecture 28 · April 4

Properties of R.e languages

(1) Union of L_1 L_2 where L_1 L_2 are r.e languages

Design of M which must accept union of L_1 and L_2

M_1 accepts L_1
M_2 accepts L_2

Multitape TM which contains the description of M_1 and also contains description of M_2. It simulates alternately M_1 and M_2 and accepts if either M_1 or M_2 accepts.

How about intersection of L_1 and L_2

Similar construction except that acceptance condition is both M_1 and M_2 accepts.

How about the case that L_1, L_2 are recursive ?

Clearly $L_1 \cap L_2$ and $L_1 \cup L_2$ are also recursive even without using alternating simulation

Also, if L is recursive $\bar{L}$ is recursive

<u>Claim</u> : If L and $\bar{L}$ are r.e. then L is recursive

<u>proof</u> : Simulate L and $\bar{L}$ alternately. At least one of the machines for L or $\bar{L}$ will halt. Report the decision

$$\boxed{\text{If } L \text{ is r.e., is } \bar{L} \text{ r.e.?}}$$

We will need more tools to address this important question

The universal language L_u

$$L_u = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$$

$$w \in \Sigma^*$$

$$w \in L(M)$$

M is the encoding of a TM M

$$M = (Q, q_0, F, \Gamma, \delta)$$

$$\delta = [\; (q, \Gamma_1, p, \Gamma_2, \{L, R\}) \;]$$

<u>Claim</u>: The ~~tape~~ alphabet of any TM can be restricted to $\{0, 1, B\}$

Claim : L_u is recursively-enumerable.

M_u simulates $\langle M \rangle$ on w

1st tape: M, w

2nd tape: contents of tape of M at any stage

current state of M

The first task of M_u is to copy w on tape 2

Then simulate M on w by consulting M's δ function as described on the first tape

Claim: L_u is r.e.

Claim: L_u is not recursive

	w_1	w_2	w_3	. .	w_j	. .
M_1	0	1	0	1	.	
M_2	1	0	1	0	0	
M_3	1	0	1			
M_4				1		
.					0	
M_i					1	

$x = i$

0/1

$$T(i,j) = \begin{cases} 1 & \text{if } w_j \in L(M_i) \\ 0 & \text{otherwise} \end{cases}$$

$$L_d = \{ x \mid x = w_i \text{ and } w_i \notin L(M_i) \}$$

Claim L_d is not r.e.

There is no TM that accepts all strings of L_d

Proof (by contradiction)

Suppose there is a TM, say M_k that accepts L_d

Then argue the behavior of M_k on w_k is not defined

M_k accepts w_k

$w_k \notin L_d$

$\therefore$ w_k should not be accepted by M_k

$\therefore$ contradiction

contradiction

M_k doesn't accept w_k

$w_k \notin L(M_k)$

$\Rightarrow$ $w_k \in L_d$

$\Rightarrow$ w_k should be accepted by M_k

contradict

Proof that L_u is not recursive

Suppose L_u is recursive. Then we can design a TM for L_d

TM for L_d

Step 1 : Given x, find $w_i = x$

(There is TM that can compute the correct i)

Step 2 : Find M_i (computable by a TM)

Step 3 : Run the "recursive" TM for L_u on $\langle M_i, w_i \rangle$

If it accepts, we reject and vice versa

Since L_d is non-r.e. the recursive TM for L_u cannot exist.

$\Rightarrow$ L_u is not recursive

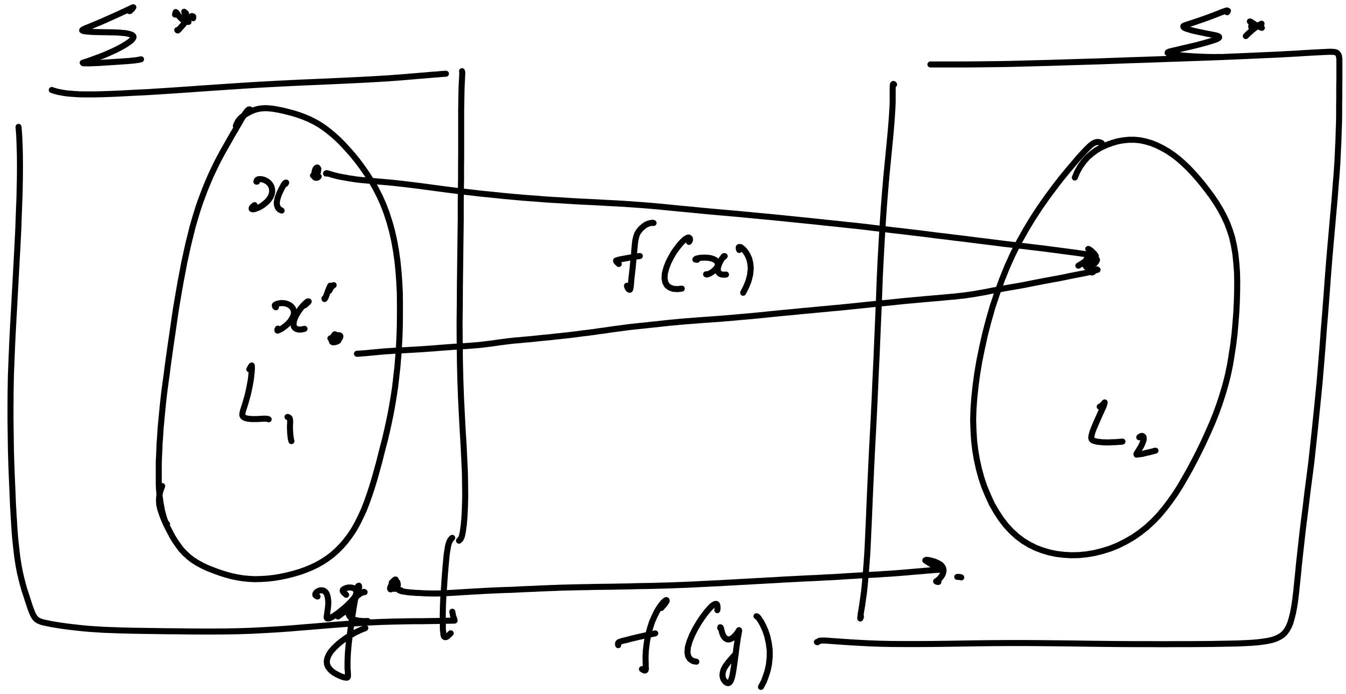
$\Rightarrow$ $\overline{L_u}$ is not r.e.

Proof by reduction

Given two languages L_1, L_2 we say that $L_1 \leq L_2$ (L_1 is reducible to L_2)

if there exists a Turing computable function $f : \Sigma^* \rightarrow \Sigma^*$

s.t. $x \in L_1$ iff $f(x) \in L_2$



many-to-one reduction

In the previous proof involving L_d and L_u

$$L_d \leq L_u$$